J Cryptol
DOI: 10.1007/s00145-017-9264-x

Journal of
CRYPTOLOGY

Characterization of Secure Multiparty Computation Without Broadcast*

Ran Cohen†
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
cohenrb@cs.biu.ac.il

Iftach Haitner‡
School of Computer Science, Tel Aviv University, Tel Aviv, Israel
iftachh@cs.tau.ac.il

Eran Omri§
Department of Computer Science and Mathematics, Ariel University, Ariel, Israel
omrier@ariel.ac.il

Lior Rotem¶
School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem, Israel
lior.rotem@cs.huji.ac.il

Communicated by Serge Fehr

Received 8 November 2015 / Revised 2 July 2017

Abstract. A major challenge in the study of cryptography is characterizing the necessary and sufficient assumptions required to carry out a given cryptographic task. The focus of this work is the necessity of a broadcast channel for securely computing symmetric functionalities (where all the parties receive the same output) when one third of the parties, or more, might be corrupted. Assuming all parties are connected via a point-to-point network, but no broadcast channel (nor a secure setup phase) is available, we prove the following characterization:

- A symmetric n-party functionality can be securely computed facing $n/3 \leq t < n/2$ corruptions (i.e., honest majority), if and only if it is $(n - 2t)$-*dominated*; a functionality is k-dominated, if *any* k-size subset of its input variables can be set to *determine* its output to some predetermined value.

*An extended abstract of this work appeared in [11].

† Ran Cohen: Work supported by THE ISRAEL SCIENCE FOUNDATION (Grant No. 189/11), the Ministry of Science, Technology and Space and by the National Cyber Bureau of Israel.

‡ Iftach Haitner: Research supported by ERC starting Grant 638121, ISF Grant 1076/11, I-CORE Grant 4/11, BSF Grant 2010196, and Check Point Institute for Information Security.

§ Eran Omri: Research supported by ISF Grant 544/13.

¶ Lior Rotem: This work was done while the author was a student at Tel Aviv University.

Published online: 22 September 2017

- Assuming the existence of one-way functions, a symmetric n-party functionality can be securely computed facing $t \geq n/2$ corruptions (i.e., no honest majority), if and only if it is 1-dominated and can be securely computed with broadcast.

It follows that, in case a third of the parties might be corrupted, broadcast is necessary for securely computing non-dominated functionalities (in which "small" subsets of the inputs cannot determine the output), including, as interesting special cases, the Boolean XOR and coin-flipping functionalities.

Keywords. Broadcast, Point-to-point communication, Multiparty computation, Coin flipping, Fairness, Impossibility result.

1. Introduction

Broadcast (introduced by Lamport et al. [24] as the Byzantine Generals problem) allows any party to deliver a message of its choice to all parties, such that all honest parties will receive the same message even if the broadcasting party is corrupted. Broadcast is an important resource for implementing secure multiparty computation. Indeed, much can be achieved when broadcast is available (hereafter, the broadcast model). In the computational setting, assuming the existence of oblivious transfer, every efficient functionality can be securely computed *with abort*,[1] facing an arbitrary number of corruptions [18,29]. Some functionalities can be computed with *full security*,[2] e.g., Boolean OR and three-party majority [20], or with $1/p$-security,[3] e.g., coin-flipping protocols [22,25]. In the information-theoretic setting, considering ideally secure communication lines between the parties, every efficient functionality can be computed with full security against unbounded adversaries,[4] facing any minority of corrupted parties [28].

The above drastically changes when broadcast or a secure setup phase are not available,[5] specifically, when considering multiparty protocols (involving more than two parties), in which the parties are connected only via a point-to-point network (hereafter, the point-to-point model) and one third of the parties, or more, might be corrupted.[6] Considering authenticated channels and assuming the existence of oblivious transfer,

[1] An efficient attack in the real world is computationally indistinguishable, via a simulator, from an attack on an "ideal computation," in which malicious parties are allowed to prematurely abort even after learning the output.

[2] The malicious parties in the "ideal computation" are *not* allowed to prematurely abort.

[3] The success probability of any efficient distinguisher between the real model and an "ideal computation" without abort is bounded from above by $1/p$.

[4] The real and ideal models are statistically close: indistinguishable even in the eyes of an all-powerful distinguisher.

[5] In case a secure setup phase is available, *authenticated broadcast* can be computed facing $t < n$ corrupted parties; Authenticated broadcast exists in the computational setting over authenticated channels assuming one-way functions exist [13] and in the information-theoretic setting over secure channels given a correlated-randomness setup [27].

[6] For two-party protocols, the broadcast model is equivalent to the point-to-point model (and thus all the results mentioned in the broadcast model hold also in the point-to-point model). If less than a third of the parties are corrupted, broadcast can be implemented using a protocol, and every functionality can be computed with information-theoretic security [3,7].

every efficient functionality can be securely computed with abort, facing an arbitrary number of corruptions [15]. In the full security model, some important functionalities *cannot* be securely computed when a third of the parties might be corrupted (e.g., Byzantine agreement [26] and three-party majority [10]), whereas other functionalities can be securely computed facing an arbitrary number of corruptions (e.g., *weak* Byzantine agreement [15] and Boolean OR [10]). The characterization of many other functionalities, however, was unknown. For instance, it was unknown whether the coin-flipping functionality or the Boolean XOR functionality can be computed with full securely, even when assuming an honest majority.

1.1. *Our Result*

A protocol is *t-consistent*, if in any execution of the protocol, in which at most t parties are corrupted, *all* honest parties output the same value. Our main technical result is the following attack on consistent protocols.

Lemma 1.1. (main lemma, informal) *Let $n \geq 3$, $t \geq \frac{n}{3}$, and let $s = n - 2t$ if $t < \frac{n}{2}$ and $s = 1$ otherwise. Let π be an n-party, t-consistent protocol in the point-to-point model with secure channels. Then, there exists an adversary that by corrupting any s-size subset $\mathcal{I}$ of the parties can do the following: first, before the execution of π, output a value $y^* = y^*(\mathcal{I})$. Second, during the execution of π, force the remaining honest parties to output y^*. The running time of the adversary is proportional to executing polynomially many instances of π.*

The lemma extends to expected polynomial-time protocols, and to protocols that only guarantee consistency to hold with high probability. We prove the lemma by extending the well-known hexagon argument of Fischer et al. [14], originally used for proving the impossibility of reaching (strong and weak) Byzantine agreement in the point-to-point model.

A corollary of Lemma 1.1 is the following lower bound on symmetric functionalities, where all parties receive the same output value. (To give a stronger lower bound, we state the result in the secure-channels model rather than in the authenticated channels model, since any functionality that can be computed with authenticated channels can also be computed with secure channels.) A functionality is *k-dominated*, if there exists a value y^* such that *any* k-size subset of the functionality input variables can be manipulated to make the output of the functionality be y^* (e.g., the Boolean OR functionality is 1-dominated with value $y^* = 1$).

Corollary 1.2. (Informal) *Let $n \geq 3$, $t \geq \frac{n}{3}$, and let $s = n - 2t$ if $t < \frac{n}{2}$ and $s = 1$ otherwise. A symmetric n-party functionality that can be computed with full security in the point-to-point model with secure channels, facing up to t corruptions, is s-dominated.*

Interestingly, the above lower bound is tight. Cohen and Lindell [10] (following Fitzi et al. [15]) showed that assuming one-way functions exist, any 1-dominated functionality (e.g., Boolean OR) that can be securely computed in the broadcast model with authenticated channels can be securely computed in the point-to-point model with authenticated channels. This shows tightness when an honest majority is not assumed. We generalize

the approach of [10], using the two-threshold detectable precomputation of Fitzi et al. [16], to get the following upper bound (in all our positive results, the efficiency of the protocol is considered to be polynomial in the circuit size of the functionality).

Proposition 1.3. (Informal) *Let $n \geq 3$ and $\frac{n}{3} \leq t < \frac{n}{2}$. Assuming up to t corruptions, any symmetric n-party functionality that is $(n-2t)$-dominated, can be computed in the secure-channels point-to-point model with information-theoretic security.*

Combining Corollary 1.2, Proposition 1.3 and [10, Thm. 7], yields the following characterization of symmetric functionalities.

Theorem 1.4. (main theorem, informal) *Let $n \geq 3$, $t \geq \frac{n}{3}$ and let f be a symmetric n-party functionality.*

1. *For $t < \frac{n}{2}$, f can be t-securely computed (with information-theoretic security) in the secure-channels point-to-point model if and only if f is $(n-2t)$-dominated.*
2. *For $t \geq \frac{n}{2}$, assuming one-way functions exist, f can be t-securely computed (with computational security) in the authenticated channels point-to-point model if and only if f is 1-dominated and can be t-securely computed (with computational security) in the authenticated channels broadcast model.*

Another application of Lemma 1.1 regards coin-flipping protocols. A coin-flipping protocol [4] allows the honest parties to jointly flip an unbiased coin, where even a coalition of (efficient) cheating parties cannot bias the outcome of the protocol by too much. We focus on protocols in which honest parties must output the same bit. Although Theorem 1.4 shows that fully secure coin flipping cannot be achieved facing one-third corruptions, we provide a stronger impossibility result under a weaker security requirement that only assumes $\frac{n}{3}$-consistency and a non-trivial bias. In particular, we show that $1/p$-secure coin flipping cannot be achieved using consistent protocols in case a third of the parties might be corrupted.

Corollary 1.5. (impossibility of multiparty coin flipping in the point-to-point model, informal) *In the secure-channels point-to-point model, there exists no $(n \geq 3)$-party coin-flipping protocol that guarantees a non-trivial bias (i.e., smaller than $\frac{1}{2}$) against an efficient adversary controlling one third of the parties.*

The above is in contrast to the broadcast model, in which coin flipping can be computed with full security if an honest majority exists [5,8], and with $1/p$-security when no honest majority is assumed [1,2,6,9,12,22].

1.2. *Our Technique*

We present the ideas underlying our main technical result, showing that the following holds in the point-to-point model. For any consistent protocol involving more than two parties, if one third of the parties (or more) might be corrupted, then there exists an adversary that can make the honest parties output a predetermined value. Furthermore, the adversary is efficient and its running time is proportional to executing a polynomial

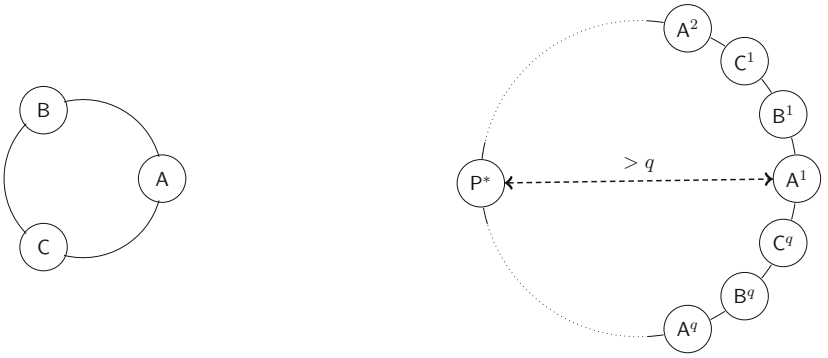

Fig. 1. Original 3-party protocol $\pi = (\mathsf{A}, \mathsf{B}, \mathsf{C})$ is on the left. On the right is the $3q$-Ring—q copies of π concatenated. Communication time between parties of opposite sides is larger than $3q/2 > q$.

number of instances of the original protocol. In the following discussion we focus on three-party protocols with a single corrupted party.

Let $\pi = (\mathsf{A}, \mathsf{B}, \mathsf{C})$ be a 1-consistent, three-party protocol, and let q be its round complexity on inputs of fixed length κ. (For the sake of clarity and by abuse of notation, we consider the joined bit-length of the deterministic inputs and random coins to be κ; see Sect. 3 for a formal analysis.) Consider the following $3q$-party protocol $\mathsf{R} = (\mathsf{A}^1, \mathsf{B}^1, \mathsf{C}^1, \ldots, \mathsf{A}^q, \mathsf{B}^q, \mathsf{C}^q)$, where the parties are connected in a ring network such that each two consecutive parties, as well as the first and last, are connected via a secure channel, and party P^j, for $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$, has the code of P (see Fig. 1).

Consider an execution of R on input $\boldsymbol{w} = (w^1_{\mathsf{A}}, w^1_{\mathsf{B}}, w^1_{\mathsf{C}}, \ldots, w^q_{\mathsf{A}}, w^q_{\mathsf{B}}, w^q_{\mathsf{C}}) \in (\{0,1\}^\kappa)^{3q}$ (i.e., party P^i has input w^i_{P}, containing its actual input and random coins – the actual inputs are arbitrarily chosen whereas the random coins are uniformly distributed). A key observation is that the view of party A^j, for instance, in this execution, is a *valid* view of the party A on input w^j_{A} in an interaction of π in which B acts honestly on input w^j_{B}. It is also a valid view of A, on input w^j_{A}, in an interaction of π in which C acts honestly on input $w^{j-1 \pmod q}_{\mathsf{C}}$. Hence, the consistency of π yields that any two consecutive parties in R output the same value, and thus *all* parties of R output the *same* value.

Consider for concreteness an attack on the parties $\{\mathsf{A}, \mathsf{B}\}$. The adversary D first selects a value $\boldsymbol{w} \in (\{0,1\}^\kappa)^{3q}$, emulates (in its head) an execution of R on $\boldsymbol{w}$, and sets y^* to be the output of the party $\mathsf{P}^* = \mathsf{A}^{q/2}$ in this execution. To interact with the parties $\{\mathsf{A}, \mathsf{B}\}$ in π, the adversary D corrupts party C and emulates an execution of R, in which all but $\{\mathsf{A}^1, \mathsf{B}^1\}$ have their inputs according to $\boldsymbol{w}$ (the roles of all parties but $\{\mathsf{A}^1, \mathsf{B}^1\}$ are played by the corrupted C), and $\{\mathsf{A}, \mathsf{B}\}$ take (without knowing it) the roles of $\{\mathsf{A}^1, \mathsf{B}^1\}$.

We claim that the output of the parties $\{\mathsf{A}, \mathsf{B}\}$ under the above attack is y^*. Observe that the emulation of R, induced by the interaction of D with $\{\mathsf{A}, \mathsf{B}\}$, is just a valid execution of R on some input $\boldsymbol{w}'$ (not completely known to the adversary). Hence, by the above observation, all parties in R (including $\{\mathsf{A}, \mathsf{B}\}$) output the same value at the end of this emulation. Since the execution of R ends after at most q rounds, and since the number of communication links between parties $\{\mathsf{A}^1, \mathsf{B}^1\}$ and the designated party P^*

is $\approx 3q/2 > q$, the actions of $\{\mathsf{A}^1, \mathsf{B}^1\}$ have *no effect* on the view of P^*. In particular, the output of P^* in the attack is also y^*, and by the above this is also the output of parties $\{\mathsf{A}, \mathsf{B}\}$.

Extension to Expected Polynomial-Round Protocolsquad The above attack works perfectly if the round complexity of π is (strictly) polynomial. For expected polynomial-round protocols, one has to work slightly harder to come up with an attack that is (almost) as good.

Let q be the expected round complexity of π. That is, an honest party of π halts after q rounds in expectation, regardless of what the other parties do, where the expectation is over its random coins. Consider the m-party protocol $\mathsf{R} = (\mathsf{A}^1, \mathsf{B}^1, \mathsf{C}^1, \ldots, \mathsf{A}^m, \mathsf{B}^m, \mathsf{C}^m)$, for $m = 2q$, connected in a ring topology as before. By Markov bound, in a random execution of R, a party halts after m rounds with probability at least $\frac{1}{2}$.

The adversary D attacking the honest parties $\{\mathsf{A}, \mathsf{B}\}$ is defined as follows. For choosing a value for y^*, it emulates an execution of R on arbitrary inputs and uniformly distributed random coins. If the party $\mathsf{P}^* = \mathsf{A}^{m/2}$ halts in at most m rounds, D sets y^* to be P^*'s output, and continues to the second stage of the attack. Otherwise, it emulates R on new inputs and random coins. Note that in k attempts, D finds a good execution with probably (at least) $1 - 2^{-k}$. After finding y^*, the adversary D continues as in the strict polynomial case discussed above.

The key observation here is that in the emulated execution of R, induced by the interaction of D with parties $\{\mathsf{A}, \mathsf{B}\}$, the party P^* *never* interacts in more than m communication rounds. Therefore, again, being far from the parties $\{\mathsf{A}, \mathsf{B}\}$, their actions do not affect P^* in the first m rounds, and so do not affect it at all. Hence, P^* outputs y^* also in the induced execution, and so do the parties $\{\mathsf{A}, \mathsf{B}\}$.

Relation of Our Attack to the Impossibility of Byzantine Agreement Our attack is based on the hexagon argument that was used to rule out three-party Byzantine agreement tolerating one corrupted party [14].[7] Assuming such a protocol exists, two copies are composed into a hexagonal system, where every pair of adjacent parties must output the same bit (by agreement) and upon starting with the same bit b, must output b, independently of the rest of the system (by validity). The adversary, controlling one party in the three-party protocol, can force the output of the remaining parties to be 0, even upon starting with input 1, by emulating toward both honest parties the hexagonal system where all virtual parties have input 0. This argument does not extend to arbitrary three-party consistent protocols, since the output of two adjacent parties might be influenced by the entire hexagonal system. We overcome this barrier by using a larger ring system, and by running the system twice: once to learn the output value and second to force the output for the honest parties in the three-party protocol.

1.3. *Additional Related Work*

Negative Results. In their seminal work, Lamport et al. [24] defined the problem of simulating a broadcast channel in the point-to-point model in terms of the Byzantine

[7] In a Byzantine agreement protocol all honest parties must output the same bit (agreement), and if all honest parties have the same input bit b, then the common output should be b (validity).

agreement problem. They showed that a broadcast protocol exists if and only if more than two-thirds of the parties are honest. Lamport [23] defined the weak Byzantine agreement problem, and showed that even this weak variant of agreement cannot be computed, using deterministic protocols, facing one-third corruptions. Fischer et al. [14] presented simpler proofs to the above impossibility results using the so-called hexagon argument, which is also the basis of our lower bound (see Sect. 1.2). They assumed a protocol exists for the three-party case, and composed multiple copies of this protocol into a ring system that contains an internal conflict. Since the ring system cannot exist, it follows that the three-party protocol does not exist. We remark that the result of [14] easily extends to public-coins protocols, where parties have access to a common random string, and even when new common random coins are revealed in each round, i.e., if the three-party protocol is defined in the coin-flipping hybrid model; this is achieved by simulating the same random coins to all instances of the three-party protocol in the ring system. It follows that coin flipping is not sufficient for solving Byzantine agreement, and thus the impossibility result for coin flipping stated in Corollary 1.5 is not implied by the aforementioned impossibility of Byzantine agreement.

Cohen and Lindell [10] analyzed the relation between security in the broadcast model and security in the point-to-point model, and showed that some (non 1-dominated) functionalities, e.g., three-party majority, that can be computed in the broadcast model cannot be securely computed in the point-to-point model, since they imply the existence of broadcast.

Positive Results If the model is augmented with a trusted setup phase, e.g., a public-key infrastructure (PKI), then broadcast can be computed facing any number of corrupted parties [24]. Pfitzmann and Waidner [27] presented an information-theoretic broadcast protocol given a correlated-randomness setup. Fitzi et al. [15] presented a probabilistic protocol that securely computes weak Byzantine agreement facing an arbitrary number of corrupted parties. Cohen and Lindell [10] showed (using the protocol from [15]) that assuming the existence of one-way functions, any 1-dominated functionality that can be securely computed in the broadcast model, can also be securely computed in the point-to-point model.

Graham and Yao [21] showed that if agreement is only required to be achieved with a constant probability (at most $(\sqrt{5}-1)/2$), then broadcast protocols exist for $n = 3$ and $t = 1$. Indeed, our attack does not apply to protocols with small consistency guarantees, and our impossibility results consider protocols that are consistent with all but negligible probability.

Goldwasser and Lindell [19] presented a weaker definition for MPC without agreement, in which non-unanimous abort is permitted, i.e., some of the honest parties may receive output while other honest parties might abort. Using this weaker definition, they utilized non-consistent protocols and constructed secure protocols in the point-to-point model, assuming an arbitrary number of corrupted parties.

1.4. *Open Questions*

Our result for the no honest majority case (second item of Theorem 1.4) requires the existence of one-way functions. In particular, given a protocol π for computing a 1-

dominated functionality f with full security in the broadcast model, one-way functions are used for compiling π into a protocol for computing f with full security in the point-to-point model.[8] It might be, however, that the existence of such a broadcast model protocol (for non-trivial functionalities) implies the existence of one-way functions, and thus adding this extra assumption is not needed.

A different interesting challenge is characterizing which *non*-symmetric functionalities can be computed in the point-to-point model, in the spirit of current work on symmetric functionalities. For example, can a three-party coin flipping in which only two parties learn the outcome coin, be computed with full security facing a single corruption?

Paper Organization

Basic definitions can be found in Sect. 2. Our attack is described in Sect. 3, and its implications are given in Sect. 4. The characterization is presented in Sect. 5.

2. Preliminaries

2.1. *Notations*

We use calligraphic letters to denote sets, uppercase for random variables, lowercase for values, boldface for vectors, and sans-serif (e.g., A) for algorithms (i.e., Turing machines). For $n \in \mathbb{N}$, let $[n] = \{1, \cdots, n\}$. Let poly denote the set all positive polynomials and let PPT denote a probabilistic (interactive) Turing machines that runs in *strictly* polynomial time. A function $\nu\colon \mathbb{N} \mapsto [0, 1]$ is *negligible*, denoted $\nu(\kappa) = \mathrm{neg}(\kappa)$, if $\nu(\kappa) < 1/p(\kappa)$ for every $p \in \mathrm{poly}$ and large enough κ. The statistical distance between two random variables X and Y over a finite set $\mathcal{U}$, denoted $\mathrm{SD}(X, Y)$, is defined as $\frac{1}{2} \cdot \sum_{u \in \mathcal{U}} |\Pr[X = u] - \Pr[Y = u]|$.

Two distribution ensembles $X = \{X(a, \kappa)\}_{a \in \{0,1\}^*, \kappa \in \mathbb{N}}$ and $Y = \{Y(a, \kappa)\}_{a \in \{0,1\}^*, \kappa \in \mathbb{N}}$ are computationally indistinguishable (denoted $X \overset{\mathrm{c}}{\equiv} Y$) if for every non-uniform polynomial-time distinguisher D there exists a function $\nu(\kappa) = \mathrm{neg}(\kappa)$, such that for every $a \in \{0, 1\}^*$ and every κ,

$$\left|\Pr\left[\mathsf{D}(X(a, \kappa), 1^\kappa) = 1\right] - \Pr\left[\mathsf{D}(Y(a, \kappa), 1^\kappa) = 1\right]\right| \leq \nu(\kappa).$$

The distribution ensembles X and Y are δ-close if for every $a \in \{0, 1\}^*$ and every κ it holds that $\mathrm{SD}(X(a, \kappa), Y(a, \kappa)) \leq \delta(\kappa)$, and statistically close (denoted $X \overset{\mathrm{s}}{\equiv} Y$) is they are δ-close and δ is negligible.

[8] For some trivial functionalities, e.g., constant functions, there exist information-theoretically secure protocols in the point-to-point model that are not based on such a compilation, and this extra assumption is not needed.

2.2. *Protocols*

We provide the basic definitions for multiparty protocols, for further details see [17]. An n-party protocol $\pi = (\mathsf{P}_1, \ldots, \mathsf{P}_n)$ is an n-tuple of probabilistic interactive Turing machines. The term *party* P_i refers to the i'th interactive Turing machine. Each party P_i starts with input $x_i \in \{0,1\}^*$ and random coins $r_i \in \{0,1\}^*$. An *adversary* D is another probabilistic interactive Turing machine describing the behavior of the corrupted parties. It starts the execution with input that contains the identities of the corrupted parties and their private inputs, and an additional auxiliary input. The parties execute the protocol in a synchronous network. That is, the execution proceeds in rounds: each round consists of a *send phase* (where parties send their message from this round) followed by a *receive phase* (where they receive messages from other parties). The adversary is assumed to be *rushing*, which means that it can see the messages the honest parties send in a round before determining the messages that the corrupted parties send in that round (we note that the attack presented in Sect. 3 actually works with weaker adversaries that do not receive an auxiliary input and are not rushing).

In the *point-to-point (communication) model*, which is the one we assume by default, all parties are connected via a *fully connected point-to-point network*. We consider two models for the communication lines between the parties: In the *authenticated channels* model, the communication lines are assumed to be ideally authenticated but not private (and thus the adversary cannot modify messages sent between two honest parties but can read them). In the *secure-channels* model, the communication lines are assumed to be ideally private (and thus the adversary cannot read or modify messages sent between two honest parties). In the *broadcast model*, all parties are given access to a physical broadcast channel in addition to the point-to-point network. In both models, no preprocessing phase is available.

Throughout the execution of the protocol, all the honest parties follow the instructions of the prescribed protocol, whereas the corrupted parties receive their instructions from the adversary. The adversary is considered to be *malicious*, meaning that it can instruct the corrupted parties to deviate from the protocol in any arbitrary way. At the conclusion of the execution, the honest parties output their prescribed output from the protocol, the corrupted parties output nothing and the adversary outputs an (arbitrary) function of its view of the computation (containing the views of the corrupted parties). The view of a party in a given execution of the protocol consists of its input, its random coins, and the messages it sees throughout this execution.

2.2.1. *Time and Round Complexity*

We start by defining both *strict* and *expected* bounds on the time complexity of interactive Turing machines (ITMs).

Definition 2.1. (*time complexity of interactive Turing machines*) An ITM P has running time T, if for every input $x \in \{0,1\}^*$ and any choice of its random coins, when interacting with arbitrary (possibly unbounded) ITMs, P's running time is at most $T(|x|)$. If $T \in \text{poly}$, then P is of (strict) polynomial time.

The machine P has an expected running time T, if for every input $x \in \{0,1\}^*$, when interacting with arbitrary (possibly unbounded) ITMs, P's expected running time, over

its own random coins, is at most $T(|x|)$. If $T \in \text{poly}$, then P has expected polynomial running time.

Next, we define both *strict* and *expected* bounds on the time and round complexities of protocols.

Definition 2.2. (*time complexity of a protocol*) Protocol $\pi = (\mathsf{P}_1, \ldots, \mathsf{P}_n)$ is a T-time protocol, if for every $i \in [n]$, party P_i has running time T. If $T \in \text{poly}$, then π is of (strict) polynomial time.

Protocol π has an expected running time T, if for every $i \in [n]$, party P_i has an expected running time T. If $T \in \text{poly}$, then π has expected polynomial running time.

Definition 2.3. (*round complexity*) Protocol $\pi = (\mathsf{P}_1, \ldots, \mathsf{P}_n)$ is a q-round protocol, if for every $i \in [n]$, and every input $x_i \in \{0,1\}^*$ and random coins $r_i \in \{0,1\}^*$, the round number in which an honest party P_i stops being active (i.e., stops sending and receiving messages) when interacting with arbitrary (possibly unbounded) interactive Turing machines $(\mathsf{P}_1^*, \ldots, \mathsf{P}_{i-1}^*, \mathsf{P}_{i+1}^*, \ldots, \mathsf{P}_n^*)$ is at most $q(|x_i|)$. If $q \in \text{poly}$, then π has (strict) polynomial round complexity.

Protocol π has an expected round complexity q, if for every $i \in [n]$, and every input $x_i \in \{0,1\}^*$, the expected round number in which an honest party P_i stops being active, over its random coins r_i, when interacting with arbitrary (possibly unbounded) interactive Turing machines $(\mathsf{P}_1^*, \ldots, \mathsf{P}_{i-1}^*, \mathsf{P}_{i+1}^*, \ldots, \mathsf{P}_n^*)$ is at most $q(|x_i|)$. If $q \in \text{poly}$, then π has expected polynomial round complexity.

Remark 2.4. Definitions 2.2 and 2.3 are fairly strong in the sense that they capture time and round complexities as local properties of every party in the protocol. Although all of our positive results readily meet these definitions, we note that weaker notions of time and round complexities can be defined, in which the running time is required to hold only when interacting with up to t corrupted parties. Our attack (Sect. 3) can be applied also to protocols with the weaker guarantees whenever $t \geq n/3$.

3. Attacking Consistent Protocols

In this section, we present a lower bound for secure protocols in the secure-channels point-to-point model. Protocols in consideration are only assumed to have a very mild security property (discussing the more standard notion of security is deferred to Sect. 4). Specifically, we only require the protocol to be consistent – all honest parties output the same value.

Definition 3.1. (*consistent protocols*) A protocol π is (δ, t)-consistent against C-class (e.g., polynomial-time, expected polynomial-time) adversaries, if the following holds. Consider an execution of π with any vector of inputs of length κ for the parties, in which a C-class adversary controls at most t parties. Then with probability at least $\delta(\kappa)$, all honest parties output the *same* value, where the probability is taken over the random coins of the adversary and of the honest parties.

In Sect. 3.1, we present an attack on consistent protocols whose round complexity is strictly bounded, and in Sect. 3.2, we extend the attack to consistent protocols with a bound on their *expected* number of rounds.

3.1. *Protocols of Strict Running Time Guarantee*

Lemma 3.2. (restating Lemma 1.1) *Let $n \geq 3$, let $t \geq \frac{n}{3}$, and let $s = n - 2t$ if $t < \frac{n}{2}$ and $s = 1$ otherwise. Let π be an n-party, T-time, q-round protocol in the secure-channels point-to-point model that is $(1-\delta, t)$-consistent against $(T_{\mathsf{D}} = 2nqT)$-time adversaries. Then, there exists a T_{D}-time adversary D such that given control over any s-size subset $\mathcal{I}$ of parties, the following holds: on security parameter κ, D first outputs a value $y^* = y^*(\mathcal{I})$. Next, D interacts with the remaining honest parties of π (i.e., with $[n] \setminus \mathcal{I}$) on arbitrary inputs of length κ unknown to D, and except for probability at most $\left(\frac{3}{2} \cdot q(\kappa) + 1\right) \cdot \delta(\kappa)$, the output of* every *honest party in this execution is y^*.*

For a polynomial-time protocol that is $(1 - \text{neg}, t)$-consistent against PPT adversaries and assuming an honest majority, Lemma 3.2 yields a PPT adversary that by controlling $n - 2t$ of the parties can manipulate the outputs of the honest parties (i.e., forcing them all to be y^*) with all but a negligible probability. If an honest majority is not assumed, the adversary can manipulate the outputs of the honest parties, by controlling any single party, except for a negligible probability. We remark that we would get slightly better parameters using an attack in which at least one honest party (but not necessarily all) outputs y^*.

We start by proving the lemma for three-party protocols, and later prove the multiparty case using a reduction to the three-party case. We actually prove a stronger statement for the three-party case, where the value y^* is independent of the set of corrupted parties. In the following lemma, we denote by T the combined running time of the parties. As opposed to T-time 3-party protocols, this more general measure captures asymmetry between the running time of the parties, and will turn out to be useful for proving Lemma 3.2.

Lemma 3.3. *(attack on three-party protocols) Let π be a 3-party, q-round protocol in the secure-channels point-to-point model, and let T be the combined running time of all three parties. If π is $(1-\delta, 1)$-consistent against $(T_{\mathsf{D}} = 2qT)$-time adversaries, then there exists a T_{D}-time adversary D such that the following holds. On security parameter κ, D first outputs a value y^*. Next, given control over* any *non-empty set of parties, D interacts with the remaining honest parties of π on arbitrary inputs of length κ unknown to D, and except for probability at most $\frac{3}{2} \cdot q(\kappa) \cdot \delta(\kappa)$, the output of* every *honest party in this execution is y^*.*

Proof. We fix the input-length parameter κ and omit it from the notation when its value is clear from the context. Let $\pi = (\mathsf{A}, \mathsf{B}, \mathsf{C})$ and let $m = q$ (assume for ease of notation that m is even). Consider, without loss of generality, that a single party is corrupted (the case of two corrupted parties follows by letting the adversary simulate an honest party) and assume for concreteness that the corrupted party is C. Consider the following $3q$-party protocol $\mathsf{R} = (\mathsf{A}^1, \mathsf{B}^1, \mathsf{C}^1, \ldots, \mathsf{A}^m, \mathsf{B}^m, \mathsf{C}^m)$, in which the parties are connected

in a ring network such that each two consecutive parties, as well as the first and last, are connected via a secure channel, and party P^j, for $\mathsf{P} \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$, has the code of P. Let $v = \kappa + T(\kappa)$, and consider an execution of R with arbitrary inputs and uniformly distributed random coins for the parties being $\boldsymbol{w} = (w_{\mathsf{A}}^1, w_{\mathsf{B}}^1, w_{\mathsf{C}}^1, \ldots, w_{\mathsf{A}}^m, w_{\mathsf{B}}^m, w_{\mathsf{C}}^m) \in (\{0,1\}^v)^{3m}$ (i.e., party P^i has input w_{P}^i, containing its actual input and random coins).

A key observation is that the point of view of the party A^j, for instance, in such an execution, is a *valid* view of the party A on input w_{A}^j in an execution of π in which B acts honestly on input w_{B}^j. It is also a valid view of A, on input w_{A}^j, in an execution of π in which C acts honestly on input $w_{\mathsf{C}}^{j-1 \pmod m}$. This observation yields the following consistency property of R.

Claim 3.4. *Consider an execution of* R *on joint input* $\boldsymbol{w} \in (\{0,1\}^v)^{3m}$*, where the parties' coins in* $\boldsymbol{w}$ *are chosen uniformly at random, and the parties' (actual) inputs are chosen arbitrarily. Then parties of distance* d *in* R*, measured by the (minimal) number of communication links between them, as well as all* $d-1$ *parties between them, output the same value with probability at least* $1 - d\delta$*.*

Proof. Consider the pair of neighboring parties $\{\mathsf{A}^j, \mathsf{B}^j\}$ in the ring R (an analogous argument holds for any two neighboring parties). Let D be an adversary, controlling the party C of π that interacts with $\{\mathsf{A}, \mathsf{B}\}$ by emulating an execution of R on $\boldsymbol{w}$ (apart from the roles of $\{\mathsf{A}^j, \mathsf{B}^j\}$), and let $\{\mathsf{A}, \mathsf{B}\}$ take (without knowing that) the roles of $\{\mathsf{A}^j, \mathsf{B}^j\}$ in this execution. The joint view of $\{\mathsf{A}, \mathsf{B}\}$ in this emulation has the same distribution as the joint view of $\{\mathsf{A}^j, \mathsf{B}^j\}$ in an execution of R. Hence, the $(1-\delta)$-consistency of π yields that A^j and B^j output the same value in an execution of R on $\boldsymbol{w}$ with probability at least $1-\delta$. The proof follows by a union bound. This concludes the proof of Claim 3.4. □

The adversary D first selects a value for $\boldsymbol{w} \in (\{0,1\}^v)^{3m}$, consisting of arbitrary input values (e.g., zeros) and uniformly distributed random coins, and sets y^* to be the output of $\mathsf{P}^* = \mathsf{A}^{m/2}$ in the execution of R on $\boldsymbol{w}$. To interact with $\{\mathsf{A}, \mathsf{B}\}$ in π, the adversary D emulates an execution of R in which all but $\{\mathsf{A}^1, \mathsf{B}^1\}$ have their inputs according to $\boldsymbol{w}$, and $\{\mathsf{A}, \mathsf{B}\}$ take the roles of $\{\mathsf{A}^1, \mathsf{B}^1\}$. The key observation is that the view of party P^* in the emulation induced by the above attack, is the *same* as its view in the execution of R on $\boldsymbol{w}$ (regardless of the inputs of $\{\mathsf{A}, \mathsf{B}\}$). This is true since the execution of R ends after at most m communication rounds. Thus, the actions of $\{\mathsf{A}, \mathsf{B}\}$ have no effect on the view of P^*, and therefore the output of P^* is y^* also in the emulated execution of R. Finally, since all the parties in the emulated execution of R have uniformly distributed random coins, and since the distance between P^* and $\{\mathsf{A}, \mathsf{B}\}$ is (less than) $\frac{3m}{2}$, Claim 3.4 yields that with probability at least $1 - \frac{3m}{2} \cdot \delta$, the output of $\{\mathsf{A}, \mathsf{B}\}$ under the above attack is y^*.

Note that the value y^* does not depend on the identity of the corrupted party, since in the first step y^* is set independently of C, and in the second step the attack follows without any change when the honest parties play the roles of $\{\mathsf{B}^1, \mathsf{C}^1\}$ if A is corrupted or $\{\mathsf{A}^2, \mathsf{C}^1\}$ if B is corrupted. □

We now proceed to prove Lemma 3.2 in the multiparty case.

Proof. Let $\pi = (\mathsf{P}_1, \ldots, \mathsf{P}_n)$ be a T-time, q-round, n-party protocol that is $(1-\delta, t)$-consistent against $2nqT$-time adversaries. We will show an adversary that by controlling any s corrupted parties, manipulates all honest parties to output a predetermined value. We separately handle the case that $\frac{n}{3} \leq t < \frac{n}{2}$ and the case that $\frac{n}{2} \leq t < n$.

Case $\frac{n}{3} \leq t < \frac{n}{2}$. Let $\mathcal{I} \subseteq [n]$ be a subset of size $s = n - 2t$, representing the indices of the corrupted parties in π. Consider the three-party protocol $\pi' = (\mathsf{A}', \mathsf{B}', \mathsf{C}')$, defined by partitioning the set $[n]$ into three subsets $\{\mathcal{I}_{\mathsf{A}'}, \mathcal{I}_{\mathsf{B}'}, \mathcal{I}\}$, where $\mathcal{I}_{\mathsf{A}'}$ and $\mathcal{I}_{\mathsf{B}'}$ are each of size t, and letting party A' run the parties $\{\mathsf{P}_i\}_{i \in \mathcal{I}_{\mathsf{A}'}}$, party B' run the parties $\{\mathsf{P}_i\}_{i \in \mathcal{I}_{\mathsf{B}'}}$ and party C' run the parties $\{\mathsf{P}_i\}_{i \in \mathcal{I}}$. Each of the parties in π' waits until all the virtual parties it is running halt, arbitrarily selects one of them and outputs the virtual party's output value.

Since the subsets $\mathcal{I}_{\mathsf{A}'}, \mathcal{I}_{\mathsf{B}'}, \mathcal{I}$ are of size at most t, the q-round, 3-party protocol π' is $(1-\delta, 1)$-consistent against $2nqT$-adversaries (otherwise there exists a $2nqT$-time adversary against the consistency of π, corrupting at most t parties). In addition, since the combined time complexity of all three parties is nT, by Lemma 3.3 there exists a $2nqT$-time adversary D' that first determines a value y^*, and later, given control over any party in π' (in particular C'), can force the two honest parties to output y^* with probability at least $1 - \frac{3q\delta}{2}$.

The attacker D for π, controlling the parties indexed by $\mathcal{I}$, is defined as follows: In the first step, D runs D' and outputs the value y^* that D' outputs. In the second step, D interacts with the honest parties in π by simulating the parties $\{\mathsf{A}', \mathsf{B}'\}$ to D', i.e., D runs D' and sends every message it receives from D' to the corresponding honest party in π, and similarly, whenever D receives a message from an honest party in π it forwards it to D'. It is immediate that there exists $i \in \mathcal{I}_{\mathsf{A}'}$ such that P_i outputs y^* in the execution of π with the same probability that A' outputs y^* in the execution of π', i.e., with probability at least $1 - \frac{3q\delta}{2}$. From the consistency property of π, all honest parties output the same value with probability at least $1 - \delta$, and using the union bound we conclude that the output of all honest parties in π under the above attack is y^* with probability at least $1 - (\frac{3q\delta}{2} + \delta)$.

Case $\frac{n}{2} \leq t < n$. Let $i^* \in [n]$ be the index of the corrupted party in π and consider the three-party protocol $\pi' = (\mathsf{A}', \mathsf{B}', \mathsf{C}')$ defined by partitioning the set $[n]$ into three subsets $\{\mathcal{I}_{\mathsf{A}'}, \mathcal{I}_{\mathsf{B}'}, \{i^*\}\}$, for $|\mathcal{I}_{\mathsf{A}'}| = \lceil \frac{n-1}{2} \rceil$ and $|\mathcal{I}_{\mathsf{B}'}| = \lfloor \frac{n-1}{2} \rfloor$. As in the previous case, the size of each subset $\mathcal{I}_{\mathsf{A}'}, \mathcal{I}_{\mathsf{B}'}, \{i^*\}$ is at most t, and the proof proceeds as above. □

3.2. *Protocols of Expected Running Time Guarantee*

In this section, we extend the attack presented above to consistent protocols with bound on their *expected* number of rounds.

Lemma 3.5. *Let $n \geq 3$, let $t \geq \frac{n}{3}$, let $s = n - 2t$ if $t < \frac{n}{2}$ and $s = 1$ otherwise, and let $z = z(\kappa)$ be an integer function. Let π be an n-party protocol of expected running time T and expected round complexity q in the secure-channels point-to-point model, that is $(1-\delta, t)$-consistent against adversaries with expected running time $T_{\mathsf{D}} = 2n(z+1)qT$. Then, there exists an adversary D with expected running time T_{D} such that given control over any s-size subset $\mathcal{I}$ of parties, the following holds: on security parameter κ, D*

first outputs a value $y^* = y^*(\mathcal{I})$. *Next,* D *interacts with the remaining honest parties of* π *on arbitrary inputs of length* κ *unknown to* D, *and except for probability at most* $2 \cdot (3 \cdot q(\kappa) + 1) \cdot \delta(\kappa) + 2^{-z(\kappa)}$, *the output of* every *honest party in this execution is* y^*.

Proof. We prove the lemma for the three-party case, the proof for the general case is similar to the proof of Lemma 3.2. We fix the input-length parameter κ and omit it from the notation when clear from the context.

Let $\pi = (\mathsf{A}, \mathsf{B}, \mathsf{C})$ and let $m = 2q$. Similarly to the proof of Lemma 3.3, we consider the (now double size) ring $\mathsf{R} = (\mathsf{A}^1, \mathsf{B}^1, \mathsf{C}^1, \ldots, \mathsf{A}^m, \mathsf{B}^m, \mathsf{C}^m)$. The attacker D follows in similar lines to those used in the proof of Lemma 3.3. The main difference is that in order to select y^*, the adversary D iterates the following for z times. In each iteration, D emulates an execution of the ring R on arbitrary inputs and uniformly distributed random coins,[9] for m communication rounds. If during one of these iterations party $\mathsf{P}^* = \mathsf{A}^{m/2}$ halts, D sets y^* to be its output in this iteration. Otherwise, in case the value y^* was not set during all z iterations, D outputs $\perp$ and aborts. The attack continues as in the proof of Lemma 3.3. □

To analyze the above attack, we first present an upper bound on the probability that D aborts.

Claim 3.6. $\Pr\left[\mathsf{D} \textit{ aborts }\right] \leq 2^{-z}$.

Proof. By Markov bound, the probability that in a single iteration of D the party P^* does not halt within $m = 2q$ rounds is at most $\frac{1}{2}$. Therefore, the probability that y^* is not set in all z iterations is at most 2^{-z}. □

Since P^* halts in the iteration that produced y^* within m rounds, its view in the emulated execution of R induced by the attack is the *same* as in this selected iteration (this holds even though some parties might run for more rounds in the emulated execution). In particular, P^* outputs y^* also in the emulated execution. The proof continues as in the proof of Lemma 3.3, where the only additional subtlety is that it is no longer true that the random coins of the parties in the emulated execution induced by the attack are uniformly distributed. Indeed, we have selected a value for $\boldsymbol{w}$ that causes P^* to halt within m rounds. Yet, since in a random execution of R, party P^* halts within m rounds with probability at least $\frac{1}{2}$, the method used to sample $\boldsymbol{w}$ at most doubles the probability of inconsistency in the ring. It follows that the attacked parties $\{\mathsf{A}, \mathsf{B}\}$ output y^* with probability at least $1 - 2 \cdot 3q\delta$ times the probability that D does not abort, and the proof of the lemma follows. □

[9] Note that now we have no a priori bound on the number of random coins used by the parties. Yet, the emulation can be done in expected time nmT.

4. Impossibility Results for Secure Computation

In this section, we present implications of the attack presented in Sect. 3 to secure multiparty computations in the secure-channels point-to-point model (note that a lower bound in the secure-channels model is stronger than in the authenticated channels model). In Sect. 4.1, we show that the only symmetric functionalities that can be securely realized, according to the real/ideal paradigm, in the presence of $n/3 \leq t < n/2$ corrupted parties (i.e., honest majority), are $(n-2t)$-dominated functionalities. The only symmetric functionalities that can be securely realized in the presence of $n/2 \leq t < n$ corrupted parties (i.e., no honest majority), are 1-dominated functionalities. In Sect. 4.2, we show that non-trivial $(n > 3)$-party coin-flipping protocols, in which the honest parties must output a bit, are impossible when facing $t \geq n/3$ corrupted parties.

For concreteness, we focus on strict polynomial-time protocols secure against strict polynomial-time adversaries, but all the results readily extend to the expected polynomial-time regime.

4.1. *Symmetric Functionalities Secure According to the Real/Ideal Paradigm*

The model of secure computation we consider is defined in Sect. 4.1.1, dominated functionalities are defined in Sect. 4.1.2, and the impossibility results are stated and proved in Sect. 4.1.3.

4.1.1. *Model Definition*

We provide the basic definitions for secure multiparty computation according to the real/ideal paradigm, for further details see [17]. Informally, a protocol is secure according to the real/ideal paradigm, if whatever an adversary can do in the real execution of protocol, can be done also in an ideal computation, in which an uncorrupted trusted party assists the computation. We consider *full security*, meaning that the ideal-model adversary cannot prematurely abort the ideal computation.

Functionalities

Definition 4.1. (*functionalities*) An n-party functionality is a random process that maps vectors of n inputs to vectors of n outputs.[10] Given an n-party functionality $f\colon (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$, let $f_i(\boldsymbol{x})$ denote its i'th output coordinate, i.e., $f_i(\boldsymbol{x}) = f(\boldsymbol{x})_i$. A functionality f is symmetric, if the output values of all parties are the same, i.e., for every $\boldsymbol{x} \in (\{0,1\}^*)^n$, $f_1(\boldsymbol{x}) = f_2(\boldsymbol{x}) = \ldots = f_n(\boldsymbol{x})$.

Real-Model Execution A real-model execution of an n-party protocol proceeds as described in Sect. 2.2.

Definition 4.2. (*real-model execution*) Let $\pi = (\mathsf{P}_1, \ldots, \mathsf{P}_n)$ be an n-party protocol and let $\mathcal{I} \subseteq [n]$ denote the set of indices of the parties corrupted by D. The joint execution of π under $(\mathsf{D}, \mathcal{I})$ in the real model, on input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$,

[10] We assume that a functionality can be computed in polynomial time.

auxiliary input z and security parameter κ, denoted $\text{REAL}_{\pi,\mathcal{I},\mathsf{D}(z)}(\boldsymbol{x},\kappa)$, is defined as the output vector of $\mathsf{P}_1,\ldots,\mathsf{P}_n$ and $\mathsf{D}(z)$ resulting from the protocol interaction, where for every $i\in\mathcal{I}$, party P_i computes its messages according to D, and for every $j\notin\mathcal{I}$, party P_j computes its messages according to π.

Ideal-Model Execution An ideal computation of an n-party functionality f on input $\boldsymbol{x}=(x_1,\ldots,x_n)$ for parties $(\mathsf{P}_1,\ldots,\mathsf{P}_n)$ in the presence of an ideal-model adversary D controlling the parties indexed by $\mathcal{I}\subseteq[n]$, proceeds via the following steps.

Sending Inputs to Trusted Party: An honest party P_i sends its input x_i to the trusted party. The adversary may send to the trusted party arbitrary inputs for the corrupted parties. Let x_i' be the value actually sent as the input of party P_i.

Trusted Party Answers the Parties: If x_i' is outside of the domain for P_i, for some index i, or if no input was sent for P_i, then the trusted party sets x_i' to be some predetermined default value. Next, the trusted party computes $f(x_1',\ldots,x_n')=(y_1,\ldots,y_n)$ and sends y_i to party P_i for every i.

Outputs: Honest parties always output the message received from the trusted party and the corrupted parties output nothing. The adversary D outputs an arbitrary function of the initial inputs $\{x_i\}_{i\in\mathcal{I}}$, the messages received by the corrupted parties from the trusted party $\{y_i\}_{i\in\mathcal{I}}$ and its auxiliary input.

Definition 4.3. (*ideal-model execution*) Let $f\colon(\{0,1\}^*)^n\mapsto(\{0,1\}^*)^n$ be an n-party functionality and let $\mathcal{I}\subseteq[n]$. The joint execution of f under (D,I) in the ideal model, on input vector $\boldsymbol{x}=(x_1,\ldots,x_n)$, auxiliary input z to D and security parameter κ, denoted $\text{IDEAL}_{f,\mathcal{I},\mathsf{D}(z)}(\boldsymbol{x},\kappa)$, is defined as the output vector of $\mathsf{P}_1,\ldots,\mathsf{P}_n$ and $\mathsf{D}(z)$ resulting from the above described ideal process.

Security Definition Having defined the real and ideal models, we can now define security of protocols according to the real/ideal paradigm.

Definition 4.4. Let $f\colon(\{0,1\}^*)^n\mapsto(\{0,1\}^*)^n$ be an n-party functionality, and let π be a probabilistic polynomial-time protocol computing f. The protocol π t-securely computes f (with computational security), if for every probabilistic polynomial-time real-model adversary D, there exists a probabilistic polynomial-time adversary S for the ideal model, such that for every $\mathcal{I}\subseteq[n]$ of size at most t, it holds that

$$\left\{\text{REAL}_{\pi,\mathcal{I},\mathsf{D}(z)}(\boldsymbol{x},\kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}}\overset{\text{c}}{\equiv}\left\{\text{IDEAL}_{f,\mathcal{I},\mathsf{S}(z)}(\boldsymbol{x},\kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}}.$$

The protocol π t-securely computes f (with information-theoretic security), if for every real-model adversary D, there exists an adversary S for the ideal model, whose running time is polynomial in the running time of D, such that for every $\mathcal{I}\subseteq[n]$ of size at most t,

$$\left\{\text{REAL}_{\pi,\mathcal{I},\mathsf{D}(z)}(\boldsymbol{x},\kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}}\overset{\text{s}}{\equiv}\left\{\text{IDEAL}_{f,\mathcal{I},\mathsf{S}(z)}(\boldsymbol{x},\kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}}.$$

4.1.2. *Dominated Functionalities*

A special class of symmetric functionalities are those with the property that every subset of a certain size can fully determine the output. For example, the multiparty Boolean AND and OR functionalities both have the property that every individual party can determine the output (for the AND functionality any party can always force the output to be 0, and for the OR functionality any party can always force the output to be 1). We distinguish between the case where there exists a single value for which every large enough subset can force the output and the case where different subsets can force the output to be different values. Looking ahead, we will focus on the first variant where a unique output value can be forced by every subset. In Sect. 4.1.3, we will show that every function that can be computed without broadcast is in fact k-dominated (where the value of k depends on the maximal number of tolerable corruptions), and in Sect. 5, we will show that dominated functions can be computed without broadcast, where the unique value (that can be forced by every large enough subset) is used as a default output value by the honest parties in case they identify some misbehavior in the protocol.

Definition 4.5. (*dominated functionalities*) A symmetric n-party functionality f is weakly k-dominated, if for every k-size subset $\mathcal{I} \subseteq [n]$ there exists a value $y^*_{\mathcal{I}}$, for which there exist inputs $\{x_i\}_{i\in\mathcal{I}}$, such that $f(x_1,\ldots,x_n) = y^*_{\mathcal{I}}$ for any complementing subset of inputs $\{x_j\}_{j\notin\mathcal{I}}$. The functionality f is k-dominated, if there exists a value y^* such that for every k-size subset $\mathcal{I} \subseteq [n]$ there exist inputs $\{x_i\}_{i\in\mathcal{I}}$, for which $f(x_1,\ldots,x_n) = y^*$ for any subset of inputs $\{x_j\}_{j\notin\mathcal{I}}$.

Example 4.6. The function $f(x_1,x_2,x_3,x_4) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$ is an example of a 4-party function that is weakly 2-dominated but not 2-dominated. Every pair of input variables can be set to determine the output value. However, there is no single output value that can be determined by all pairs, for example, $\{x_1, x_2\}$ can force the output to be 1 (by setting $x_1 = x_2 = 1$) whereas $\{x_1, x_3\}$ can force the output to be 0 (by setting $x_1 = x_3 = 0$). The function

$$f_{\text{2-of-4}}(x_1,x_2,x_3,x_4) = (x_1\wedge x_2)\vee(x_1\wedge x_3)\vee(x_1\wedge x_4)\vee(x_2\wedge x_3)\vee(x_2\wedge x_4)\vee(x_3\wedge x_4)$$

is 2-dominated with value $y^* = 1$.

Claim 4.7. *Let f be an n-party functionality and let $m \leq \frac{n}{3}$. If f is weakly m-dominated, then it is m-dominated.*

Proof. Let $\mathcal{I}_1, \mathcal{I}_2 \subseteq [n]$ be two subsets of size m. Denote by $\{x_i\}_{i\in\mathcal{I}_1}$ (resp., $\{x_i\}_{i\in\mathcal{I}_2}$) the input values for $\mathcal{I}_1$ (resp., $\mathcal{I}_2$) that force the output to be $y^*_{\mathcal{I}_1}$ (resp., $y^*_{\mathcal{I}_2}$). In case $\mathcal{I}_1$ and $\mathcal{I}_2$ are disjoint, fix an arbitrary complementing subset of inputs $\{x_j\}_{j\notin\mathcal{I}_1\cup\mathcal{I}_2}$. On the one hand it holds that $f(x_1,\ldots,x_n) = y^*_{\mathcal{I}_1}$ and on the other hand it holds that $f(x_1,\ldots,x_n) = y^*_{\mathcal{I}_2}$, hence $y^*_{\mathcal{I}_1} = y^*_{\mathcal{I}_2}$.

In case $\mathcal{I}_1$ and $\mathcal{I}_2$ are not disjoint, it holds that $|\mathcal{I}_1 \cup \mathcal{I}_2| < 2m \leq \frac{2n}{3}$ and since $m \leq \frac{n}{3}$, there exists a subset $\mathcal{I}_3 \subseteq [n] \setminus (\mathcal{I}_1 \cup \mathcal{I}_2)$ of size m. Denote by $y^*_{\mathcal{I}_3}$ the output value that can be determined by the input variables $\{x_i\}_{i\in\mathcal{I}_3}$ ($y^*_{\mathcal{I}_3}$ is guaranteed to exist since

f is weakly m-dominated). $\mathcal{I}_3$ is disjoint from $\mathcal{I}_1$ and from $\mathcal{I}_2$, so it follows from the argument above that $y^*_{\mathcal{I}_1} = y^*_{\mathcal{I}_3}$ and $y^*_{\mathcal{I}_2} = y^*_{\mathcal{I}_3}$, therefore $y^*_{\mathcal{I}_1} = y^*_{\mathcal{I}_2}$. □

4.1.3. *The Lower Bound*

Lemma 4.8. (restating Corollary 1.2) *Let $n \geq 3$, let $t \geq \frac{n}{3}$ and let f be a symmetric n-party functionality that can be t-securely computed in the secure-channels point-to-point model with computational security.*

1. *If $\frac{n}{3} \leq t < \frac{n}{2}$, then f is $(n-2t)$-dominated.*
2. *If $\frac{n}{2} \leq t < n$, then f is 1-dominated.*

Proof. Assume that $\frac{n}{3} \leq t < \frac{n}{2}$ (the proof for $\frac{n}{2} \leq t < n$ is similar). Let π be a protocol that t-securely computes f in the point-to-point model with secure channels. Since f is symmetric, all honest parties output the same value (except for a negligible probability), hence π is $(1 - \text{neg}, t)$-consistent; let D be the PPT adversary guaranteed from Lemma 3.2 and let $\mathcal{I} \subseteq [n]$ be any subset of size $n - 2t$. It follows that given control over $\{\mathsf{P}_i\}_{i\in\mathcal{I}}$, the adversary D can first fix a value $y^*_{\mathcal{I}}$, and later force the output of the honest parties to be $y^*_{\mathcal{I}}$ (except for a negligible probability). Since π is a protocol that t-securely computes f and $n - 2t \leq t$, there exists an ideal-model adversary S that upon corrupting $\{\mathsf{P}_i\}_{i\in\mathcal{I}}$ can force the output of the honest parties in the ideal-model computation to be $y^*_{\mathcal{I}}$. All S can do is to select the input values of the corrupted parties, hence, there must exist input values $\{x_i\}_{i\in\mathcal{I}}$ that determine the output of the honest parties to be $y^*_{\mathcal{I}}$, i.e., f is weakly $(n-2t)$-dominated. Since $n - 2t \leq \frac{n}{3}$ and following Claim 4.7 we conclude that f is $(n-2t)$-dominated. □

4.2. *Coin-Flipping Protocols*

A coin-flipping protocol [4] allows the honest parties to jointly flip an unbiased coin, where even a coalition of (efficient) cheating parties cannot bias the outcome of the protocol by much. Our focus is on coin flipping, where the honest parties *must* output a bit. Although Lemma 4.8 immediately shows that coin flipping cannot be securely computed according to the real/ideal paradigm, we present a stronger impossibility result by considering weaker security requirements. For simplicity, we consider coin-flipping protocols with perfect consistency; however, our negative result readily extends to protocols where consistency is only guaranteed to hold with high probability.

Definition 4.9. (*coin-flipping protocol*) A polynomial-time n-party protocol π is a (γ, t)-bias coin-flipping protocol, if the following holds.

1. π is $(1, t)$-consistent against PPT adversaries.
2. When interacting on security parameter κ with a PPT adversary controlling at most t corrupted parties, the common output of the honest parties is $\gamma(\kappa)$-close to being a uniform bit. (In particular, the honest parties are allowed to output $\perp$, or values other than $\{0, 1\}$, with probability at most γ.)

The following is a straightforward application of Lemma 3.2.

Lemma 4.10. (restating Corollary 1.5) *In the secure-channels point-to-point model, for $n \geq 3$ and $\gamma(\kappa) < \frac{1}{2} - 2^{-\kappa}$, there exists no n-party, $(\gamma, \lceil \frac{n}{3} \rceil)$-bias coin-flipping protocol.*

Proof. Let π be a point-to-point n-party $(\gamma, \lceil \frac{n}{3} \rceil)$-bias coin-flipping protocol. Let D be the PPT adversary that is guaranteed by Lemma 3.2 (since π is $(1, \lceil \frac{n}{3} \rceil)$-consistent against PPT adversaries). Consider some fixed set of $\lceil \frac{n}{3} \rceil$ corrupted parties of π and let $Y(\kappa)$ denote the random variable of $\mathsf{D}(\kappa)$'s output in the first step of the attack. Without loss of generality, for infinitely many values of κ it holds that $\Pr[Y(\kappa) = 0] \leq \frac{1}{2}$. Consider the adversary D' that on security parameter κ, repeats the first step of $\mathsf{D}(\kappa)$ until the resulting value of y^* is nonzero or κ failed attempts have been reached, where if the latter happens D' aborts. Next, D' continues the nonzero execution of D to make the honest parties of π output y^*. It is immediate that for infinitely many values of κ, the common output of the honest parties under the above attack is 0 with probability at most $2^{-\kappa}$, and hence the common output of the honest parties is $\frac{1}{2} - 2^{-\kappa}$ far from uniform. Thus, π is not a $(\gamma, \lceil \frac{n}{3} \rceil)$-bias coin-flipping protocol. □

5. Characterizing Secure Computation Without Broadcast

In this section, we show that the lower bounds presented in Lemma 4.8 are tight. We treat separately the case where an honest majority is assumed and the case where no honest majority is assumed.

5.1. *No Honest Majority*

Cohen and Lindell [10, Thm. 7] showed that any 1-dominated functionality that can be t-securely computed in the broadcast model, can also be t-securely computed in the point-to-point model. The idea in [10] is based on the protocols of Fitzi et al. [15, Thm. 2 and 3] that are secure with (fair) abort and compute correlated-randomness to be used for authenticated Byzantine agreement protocols. Fitzi et al. [15] constructed two such protocols: the first protocol is computationally secure in the authenticated channels model under the assumption that one-way functions exist, and the second protocol is information-theoretically secure in the secure-channels model. With the goal of constructing no honest-majority MPC protocols, Cohen and Lindell [10] focused on the computational setting, and using the protocol in [15, Thm. 2] proved the above statement in the authenticated channels model assuming one-way functions exist. An information-theoretic variant in the secure-channels model follows in a similar way using [15, Thm. 3].

Theorem 5.1. ([10, Thm. 7]) *Let $n \geq 3$, let $t < n$, and let f be a 1-dominated functionality.*

1. *Assuming the existence of one-way functions, if f can be t-securely computed in the authenticated channels broadcast model then f can be t-securely computed in the authenticated channels point-to-point model.*
2. *If f can be t-securely computed in the secure-channels broadcast model then f can be t-securely computed in the secure-channels point-to-point model.*

Combining with Lemma 4.8, we establish the following result.

Theorem 5.2. (restating second part of Theorem 1.4) *Let $n \geq 3$, let $\frac{n}{2} \leq t < n$, and assume that one-way functions exist. An n-party functionality can be t-securely computed in the authenticated channels point-to-point model, if and only if it is 1-dominated and can be t-securely computed in the authenticated channels broadcast model.*

Proof. Immediately by Lemma 4.8 and the first item of Theorem 5.1. □

5.2. *Honest Majority*

To prove the matching upper bound in the honest-majority setting (Proposition 5.6 below) we use the *two-threshold multiparty protocol* of Fitzi et al. [16, Thm. 6]. This protocol with parameters t_1, t_2 runs in the point-to-point model with secure channels, and whenever $t_1 \leq t_2$ and $t_1 + 2t_2 < n$, the following holds. Let $\mathcal{I}$ be the set of parties that the (computationally unbounded) adversary corrupts. If $|\mathcal{I}| \leq t_1$, then the protocol computes f with full security. If $t_1 < |\mathcal{I}| \leq t_2$, then the protocol securely computes f with fairness (i.e., the adversary may force *all* honest parties to output $\perp$, provided that it learns no new information). In Sect. 5.2.1, we formally define the notion of two-threshold security that captures the security achieved by the protocol of Fitzi et al. [16]. In Sect. 5.2.2, we present the fully secure protocol for $(n-2t)$-dominated functionalities.

5.2.1. *Defining Two-Threshold Security*

We present a weaker variant of the ideal model that allows for a premature (and fair) abort, in case sufficiently many parties are corrupted. Next, we define two-threshold security of protocols.

Threshold ideal-model execution A t-threshold ideal computation of an n-party functionality f on input $\boldsymbol{x} = (x_1, \ldots, x_n)$ for parties $(\mathsf{P}_1, \ldots, \mathsf{P}_n)$, in the presence of an ideal-model adversary D controlling the parties indexed by $\mathcal{I} \subseteq [n]$, proceeds via the following steps.

Sending Inputs to Trusted Party: An honest party P_i sends its input x_i to the trusted party. The adversary may send to the trusted party arbitrary inputs for the corrupted parties. If $|\mathcal{I}| > t$, then the adversary may send a special abort command to the trusted party. Let x_i' be the value actually sent as the input of party P_i.

Trusted Party Answers the Parties: If the adversary sends the special abort command (specifically, $|\mathcal{I}| > t$), then the trusted party sends $\perp$ to all the parties. Otherwise, if x_i' is outside of the domain for P_i, for some index i, or if no input is sent for P_i, then the trusted party sets x_i' to be some predetermined default value. Next, the trusted party computes $f(x_1', \ldots, x_n') = (y_1, \ldots, y_n)$ and sends y_i to party P_i for every i.

Outputs: Honest parties always output the message received from the trusted party and the corrupted parties output nothing. The adversary D outputs an arbitrary function of the initial inputs $\{x_i\}_{i \in \mathcal{I}}$, the messages received by the corrupted parties from the trusted party $\{y_i\}_{i \in \mathcal{I}}$ and its auxiliary input.

Definition 5.3. (*Threshold ideal-model computation*) Let $f\colon (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$ be an n-party functionality and let $\mathcal{I} \subseteq [n]$. The joint execution of f under (D, I) in the t-threshold ideal model, on input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, auxiliary input z to D and security parameter κ, denoted $\text{IDEAL}^{t}_{f,\mathcal{I},\mathsf{D}(z)}(\boldsymbol{x}, \kappa)$, is defined as the output vector of $\mathsf{P}_1, \ldots, \mathsf{P}_n$ and $\mathsf{D}(z)$ resulting from the above described ideal process.

Definition 5.4. Let $f\colon (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$ be an n-party functionality, and let π be a probabilistic polynomial-time protocol computing f. The protocol π (t_1, t_2)-securely computes f (with information-theoretic security), if for every real-model adversary D, there exists an adversary S for the t_1-threshold ideal model, whose running time is polynomial in the running time of D, such that for every $\mathcal{I} \subseteq [n]$ of size at most t_2

$$\left\{\text{REAL}_{\pi,\mathcal{I},\mathsf{D}(z)}(\boldsymbol{x}, \kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}} \overset{s}{\equiv} \left\{\text{IDEAL}^{t_1}_{f,\mathcal{I},\mathsf{S}(z)}(\boldsymbol{x}, \kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}}.$$

Theorem 5.5. ([16, Thm. 6]) *Let $n \geq 3$, let t_1, t_2 be parameters such that $t_1 \leq t_2$ and $t_1 + 2t_2 < n$, and let f be an n-party functionality. Then, f can be (t_1, t_2)-securely computed in the secure-channels point-to-point model with information-theoretic security.*

5.2.2. *Full Security with an Honest Majority*

Proposition 5.6. *Let $n \geq 3$, let $\frac{n}{3} \leq t < \frac{n}{2}$, and let f be a symmetric n-party functionality. If f is $(n-2t)$-dominated, then it can be t-securely computed in the secure-channels point-to-point model with information-theoretic security.*

Proof. Let f be an $(n-2t)$-dominated functionality with default output value y^*. If $n - 2t = 1$, then f is 1-dominated, and since $t < \frac{n}{2}$, f can be t-securely computed with information-theoretic security in the secure-channels broadcast model (e.g., using Rabin and Ben-Or [28]). Hence, the proposition follows from the second item of Theorem 5.1.

For $n - 2t \geq 2$, set $t_1 = n - 2t - 1$ and $t_2 = t$, and let π' be the n-party protocol, guaranteed to exist by Theorem 5.5, that (t_1, t_2)-securely computes f. We define π to be the following n-party protocol for computing f in the point-to-point model with secure channels.

Protocol 5.7. *(full security for $(n-2t)$-dominated functionalities)*

1. *The parties run the protocol π'. Let y_i be the output of P_i at the end of the execution.*
2. *If $y_i \neq \perp$, party P_i outputs y_i, otherwise it outputs y^*.*

Let D be an adversary attacking the execution of π and let $\mathcal{I} \subseteq [n]$ be a subset of size at most t. It follows from Theorem 5.5 that there exists a (possibly aborting) adversary S' for D in the t_1-threshold ideal model such that

$$\left\{\text{REAL}_{\pi',\mathcal{I},\mathsf{D}(z)}(\boldsymbol{x}, \kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}} \overset{s}{\equiv} \left\{\text{IDEAL}^{t_1}_{f,\mathcal{I},\mathsf{S}'(z)}(\boldsymbol{x}, \kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}}.$$

Using S', we construct the following non-aborting adversary S for the full security ideal model. On inputs $\{x_i\}_{i\in\mathcal{I}}$ and auxiliary input z, S starts by emulating S' on these inputs, playing the role of the trusted party (in the t_1-threshold ideal model). If S' sends an abort command, it is guaranteed that $|\mathcal{I}| \geq n-2t$ and since f is $(n-2t)$-dominated, there exist input values $\{x'_i\}_{i\in\mathcal{I}}$ that determine the output of f to be y^*. So in this case, S sends these $\{x'_i\}_{i\in\mathcal{I}}$ to the trusted party (in the full security ideal model) and returns $\perp$ to S'. Otherwise, S' does not abort and S forwards the message from S' to the trusted party and the answer from the trusted party back to S'. In both cases S outputs whatever S' outputs and halts.

A main observation is that the views of the adversary D in an execution of π and in an execution of π' (with the same inputs and random coins) are identical. This holds since the only difference between π and π' is in the second step of π that does not involve any interaction. It follows that in case the output of the parties in Step 1 of π is not $\perp$, the joint distribution of the honest parties' output and the output of D in π is statistically close to the output of the honest parties and of S in the full security ideal model (since the latter is exactly the output of the honest parties and of S' in the t_1-threshold ideal model). If the output in Step 1 of π is $\perp$, then all honest parties in π output y^*. In this case S' sends abort (except for a negligible probability) and since S sends to the trusted party the input values $\{x'_i\}_{i\in\mathcal{I}}$ that determine the output of f to be y^*, the honest parties' output is y^* also in the ideal computation. We conclude that

$$\left\{\mathrm{REAL}_{\pi,\mathcal{I},\mathsf{D}(z)}(\boldsymbol{x},\kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}} \overset{\mathrm{s}}{\equiv} \left\{\mathrm{IDEAL}_{f,\mathcal{I},\mathsf{S}(z)}(\boldsymbol{x},\kappa)\right\}_{(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1},\kappa\in\mathbb{N}}.$$

□

Theorem 5.8. (restating the first part of Theorem 1.4) *Let $n \geq 3$ and $\frac{n}{3} \leq t < \frac{n}{2}$. A symmetric n-party functionality can be t-securely computed in the secure-channels point-to-point model, if and only if it is $(n-2t)$-dominated.*

Proof. Immediately follows by Lemma 4.8 and Proposition 5.6. □

References

[1] B. Alon, E. Omri. Almost-optimally fair multiparty coin-tossing with nearly three-quarters malicious, in *Proceedings of the 14th Theory of Cryptography Conference, TCC 2016-B, part I*. (2016) pp. 307–335

[2] A. Beimel, E. Omri, I. Orlov. Protocols for multiparty coin toss with a dishonest majority. *Journal of Cryptology*. **28**(3), 551–600 (2015)

[3] M. Ben-Or, S. Goldwasser, A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract), in *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS)*. (1988) pp. 1–10

[4] M. Blum. Coin flipping by telephone, in *Advances in Cryptology—CRYPTO '81*. (1981) pp. 11–15

[5] A.Z. Broder, D. Dolev. Flipping coins in many pockets (Byzantine agreement on uniformly random values), in *Proceedings of the 25th Annual Symposium on Foundations of Computer Science (FOCS)*. (1984) pp. 157–170

[6] N. Buchbinder, I. Haitner, N. Levi, E. Tsfadia. Fair coin flipping: Tighter analysis and the many-party case, in *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. (2017) pp. 2580–2600

[7] D. Chaum, C. Crépeau, I. Damgård. Multiparty unconditionally secure protocols (extended abstract), in *Proceedings of the 10th Annual ACM Symposium on Theory of Computing (STOC)*. (1988) pp. 11–19
[8] B. Chor, S. Goldwasser, S. Micali, B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract), in *Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS)*. (1985) pp. 383–395
[9] R. Cleve. Limits on the security of coin flips when half the processors are faulty, in *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*. (1986) pp. 364–369
[10] R. Cohen, Y. Lindell. Fairness versus guaranteed output delivery in secure multiparty computation, in *Advances in Cryptology—ASIACRYPT 2014, part II*. (2014) pp. 466–485
[11] R. Cohen, I. Haitner, E. Omri, L. Rotem. Characterization of secure multiparty computation without broadcast, in *Proceedings of the 13th Theory of Cryptography Conference, TCC 2016-A, part I*. (2016) pp. 596–616
[12] R. Cohen, I. Haitner, E. Omri, L. Rotem. From fairness to full security in multiparty computation. Manuscript. (2017)
[13] D. Dolev, R. Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*. **12**(4), 656–666, (1983)
[14] M.J. Fischer, N.A. Lynch, M. Merritt. Easy impossibility proofs for distributed consensus problems, in *Proceedings of the Fourth Annual ACM Symposium on Principles of Distributed Computing (PODC)*. (1985) pp. 59–70
[15] M. Fitzi, D. Gottesman, M. Hirt, T. Holenstein, A. Smith. Detectable Byzantine agreement secure against faulty majorities, in *Proceedings of the 21st Annual ACM Symposium on Principles of Distributed Computing (PODC)*. (2002) pp. 118–126
[16] M. Fitzi, M. Hirt, T. Holenstein, J. Wullschleger. Two-threshold broadcast and detectable multi-party computation, in *Advances in Cryptology—EUROCRYPT 2003*. (2003) pp. 51–67
[17] O. Goldreich. *Foundations of Cryptography—VOLUME 2: Basic Applications*. Cambridge University Press, (2004)
[18] O. Goldreich, S. Micali, A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority, in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*. (1987) pp. 218–229
[19] S. Goldwasser, Y. Lindell. Secure multi-party computation without agreement. *Journal of Cryptology*. **18**(3), 247–287 (2005)
[20] D. Gordon, J. Katz. Complete fairness in multi-party computation without an honest majority, in *Proceedings of the 6th Theory of Cryptography Conference, TCC 2009*. (2009) pp. 19–35
[21] R.L. Graham, A.C. Yao. On the improbability of reaching byzantine agreements (preliminary version), in *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC)*. (1989) pp. 467–478
[22] I. Haitner, E. Tsfadia. An almost-optimally fair three-party coin-flipping protocol, in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*. (2014) pp. 817–836
[23] L. Lamport. The weak Byzantine generals problem. *Journal of the ACM*. **30**(3), 668–676 (1983)
[24] L. Lamport, R. E. Shostak, M. C. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*. **4**(3), 382–401 (1982)
[25] T. Moran, M. Naor, G. Segev. An optimally fair coin toss, in *Proceedings of the 6th Theory of Cryptography Conference, TCC 2009*. (2009) pp. 1–18
[26] M. C. Pease, R. E. Shostak, L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*. **27**(2), 228–234 (1980)
[27] B. Pfitzmann, M. Waidner. Unconditional Byzantine agreement for any number of faulty processors, in *Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*. (1992) pp. 339–350
[28] T. Rabin, M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract), in *Proceedings of the 30th Annual Symposium on Foundations of Computer Science (FOCS)*. (1989) pp. 73–85
[29] A.C. Yao. Protocols for secure computations, in *Proceedings of the 23th Annual Symposium on Foundations of Computer Science (FOCS)*. (1982) pp. 160–164